Tailoring Application Performance Across the Abstraction Layers

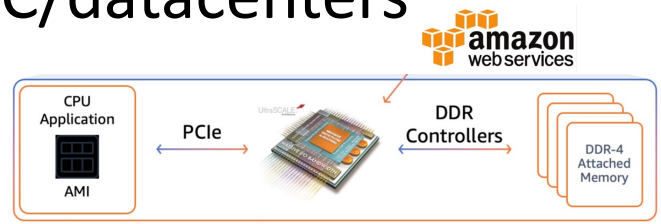
Georgi Gaydadjiev
Computer Engineering
TU Delft

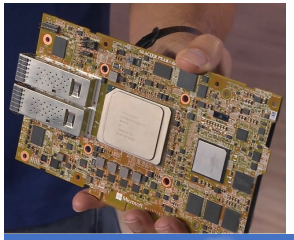
May 1st, 2025
HPC & The Roaring 20s of Computing
VU, Amsterdam, the Netherlands

TUDelft

Motivation for Reconfigurable Accelerators

- Ongoing interest in using FPGAs in HPC/datacenters
  - Key advantage: Performance/Watt



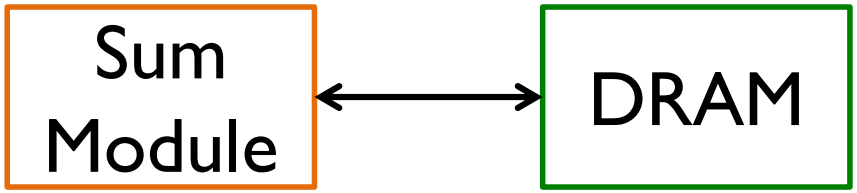


- Some key domains:
  - Big data analytics, scientific computing, image processing, genomics, financial analytics, search, network functions, etc

Problem: Programmability

- Verilog and VHDL **too low level** for software developers
- High level synthesis (HLS) tools need user pragmas to help "discover" parallelism
  - C-based input, pragmas **requiring hardware knowledge**
  - Limited in exploiting data locality
  - Difficult to synthesize complex data paths with nested parallelism

Example: Vivado HLS



Add 512 integers stored in external DRAM

```
void(int* mem) {

    mem[512] = 0;

    for(int i=0; i<512; i++) {
        mem[512] += mem[i];
    }
}
```

27,236 clock cycles for computation

Two-orders of magnitude too long!

Optimized Design: Vivado HLS

```
#define CHUNKSIZE (sizeof(MPort)/sizeof(int))
#define LOOPCOUNT (512/CHUNKSIZE)

void(MPort* mem) {
    MPort buff[LOOPCOUNT];
    memcpy(buff, mem, LOOPCOUNT);

    int sum = 0;
    for(int i=1; i<LOOPCOUNT; i++) {
        #pragma PIPELINE
        for(int j=0; j<CHUNKSIZE; j++) {
        #pragma UNROLL
        sum += (int)(buff[i]>>j*sizeof(int)*8);
    }
    }
    mem[512] = sum;
}
```

Width of DRAM controller interface

Burst Access

Use local variable

Loop Restructuring

Special compiler directives

Bit shifting to extract individual elements

302 clock cycles for computation

Does this remind us of something?

```
ARCHITECTURE behavioral OF JetSum IS
  SIGNAL jetTmp : tJet := cEmptyJet;
  SIGNAL EnergyTmp , EcalTmp :
STD_LOGIC_VECTOR( 21 DOWNTO 0 );
BEGIN
  PROCESS( clk )
  BEGIN
    IF( RISING_EDGE( clk ) ) THEN
      IF( NOT jetIn1.DataValid ) THEN
        jetOut <= cEmptyJet;
      ELSE
        jetOut.Energy( 15 DOWNTO 0 ) <=
            jetIn1.Energy + jetIn2.Energy;
        jetOut.Ecal( 15 DOWNTO 0 )   <=
             jetIn1.Ecal + jetIn2.Ecal;
        jetOut.DataValid <= TRUE;
      END IF;
    END IF;
  END PROCESS;
END ARCHITECTURE behavioral;
```

- **Would something like this be better?**

```
class Jet extends KernelLib{
 //… constructors etc
 public static Jet add(Jet lJet, Jet rJet) {
   DFEVar energy = lJet.energy() + rJet.egergy();
   DFEVar ecal = lJet.ecal() + rJet.ecal();
   DFEVar valid = lJet.valid() + rJet.valid();
   return new Jet(lJet.kernel(), energy, ecal, valid);
 }
}
```

real example from The CMS Experiment at CERN

- MaxJ[1] code is just Software
  - running it generates the VHDL
- No need to keep track on exact bit sizes, fixed point positions, etc
- Other kernels benefit even more
  - **Bitonic Sort** is
  - 500 VHDL lines versus only 130 in MaxJ

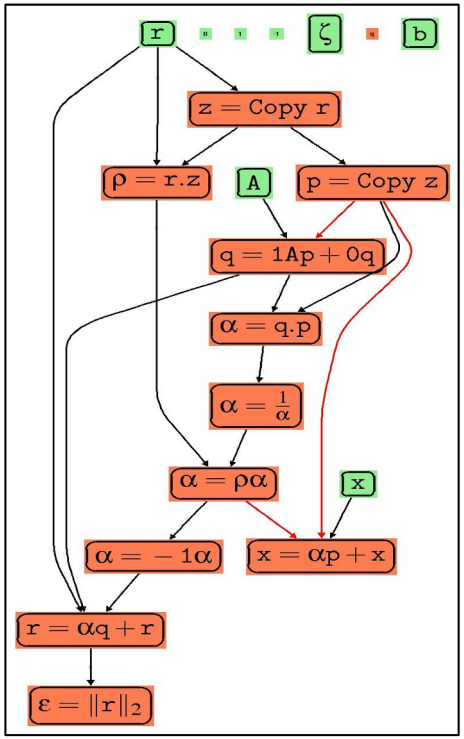
[1] – © Maxeler Technologies, a Groq company

So, for programmable FPGAs we need to ...

- Define ***Higher-level Abstractions*** than HLS to
  - *Enhance Productivity*: developers focus on application
  - *Direct Performance Control*:
    - ***Capture Locality*** to reduce off-chip memory traffic
    - ***Exploit Multi-scale Parallelism*** at different levels

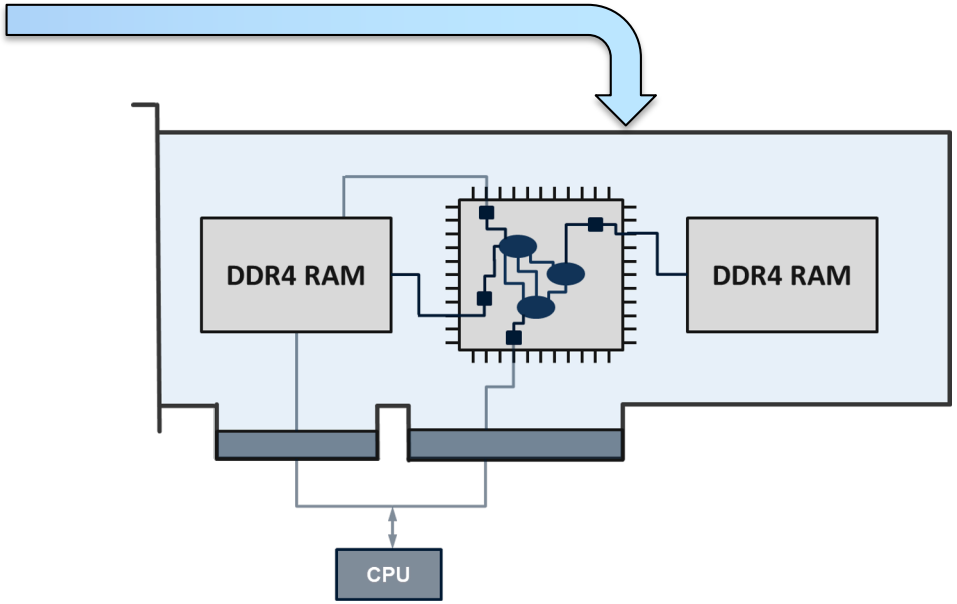
- Clear Programming and Execution Models
  - streaming dataflow on **fat** systolic kernels
  - fully customisable datatypes
- Powerful tools to facilitate the rapid development
  - Expose key platform-specific locality/parallelism properties
  - Provide library functions (parallel patterns, type casts and more)
- Dedicated compiler to generate efficient hardware
- Linker and runtime system Linux integration
- **Open source** is a key to remove usage barriers

From CG Algorithm to Dataflow Hardware



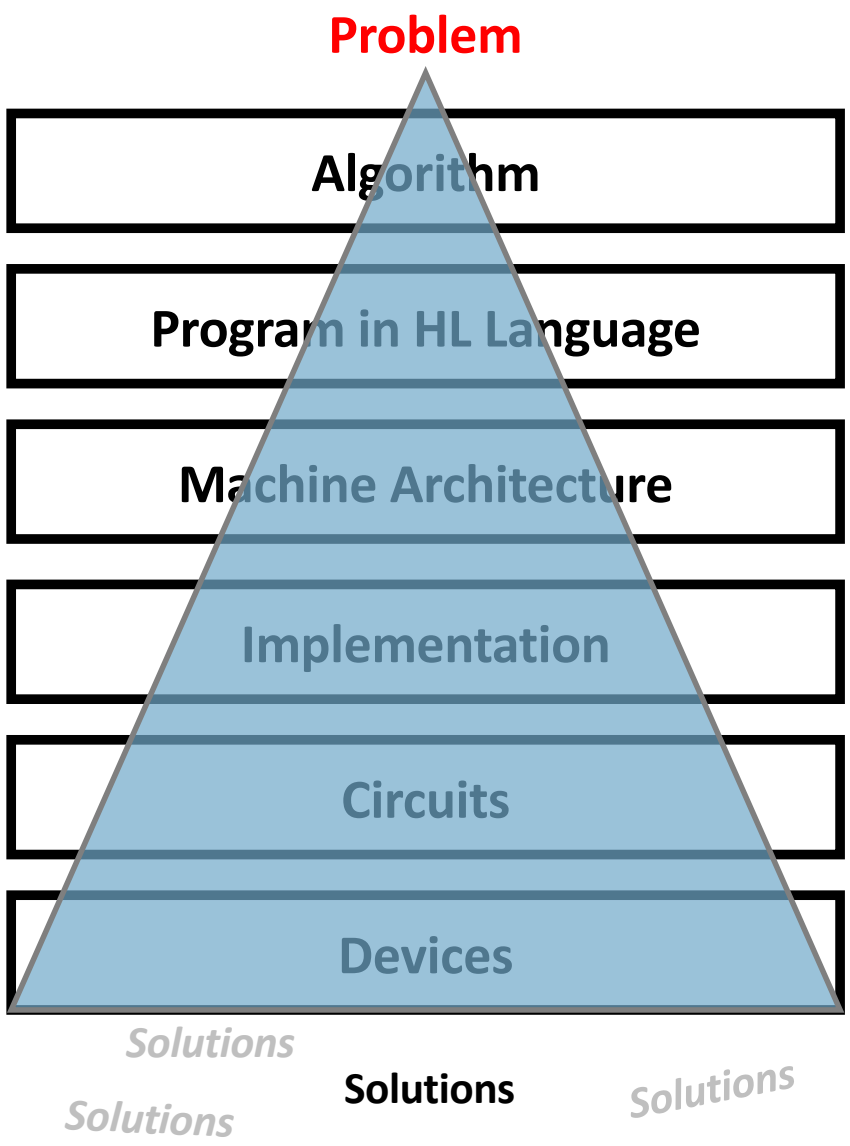
1. Describe Conjugate Gradient as dataflow graph

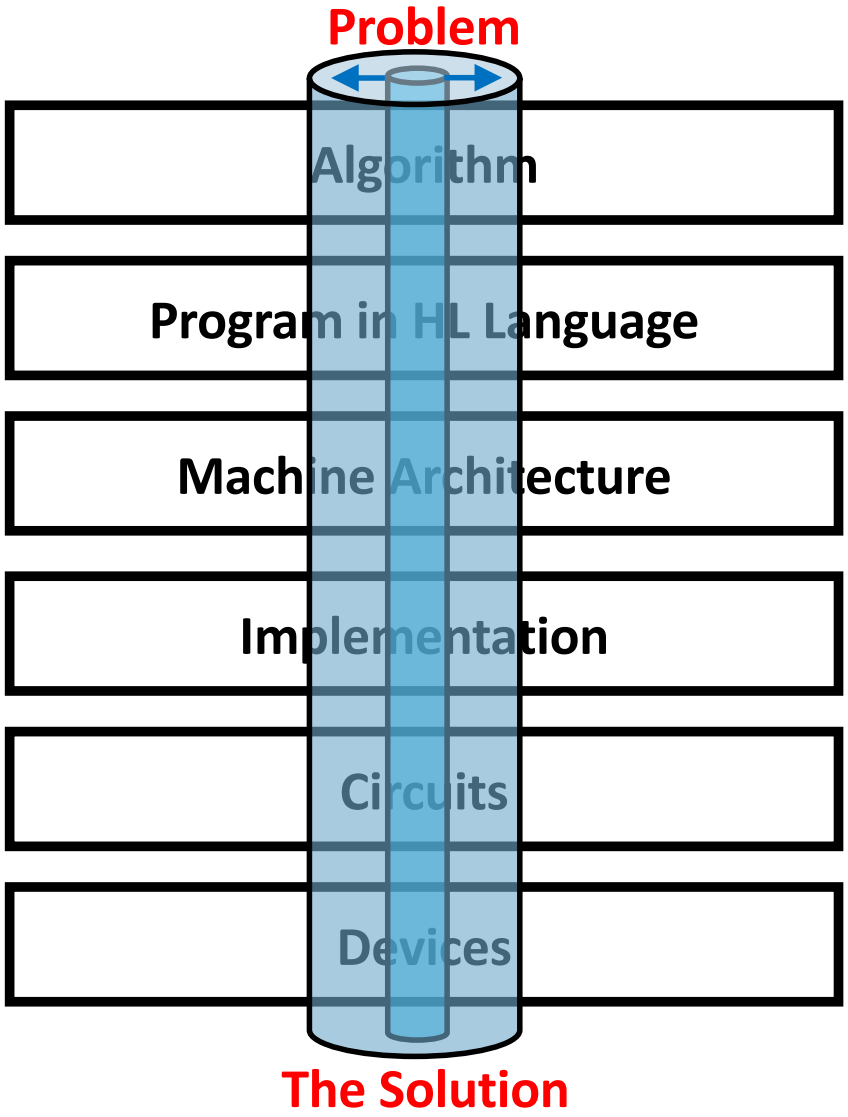
2. Compile dataflow structure and load to hardware



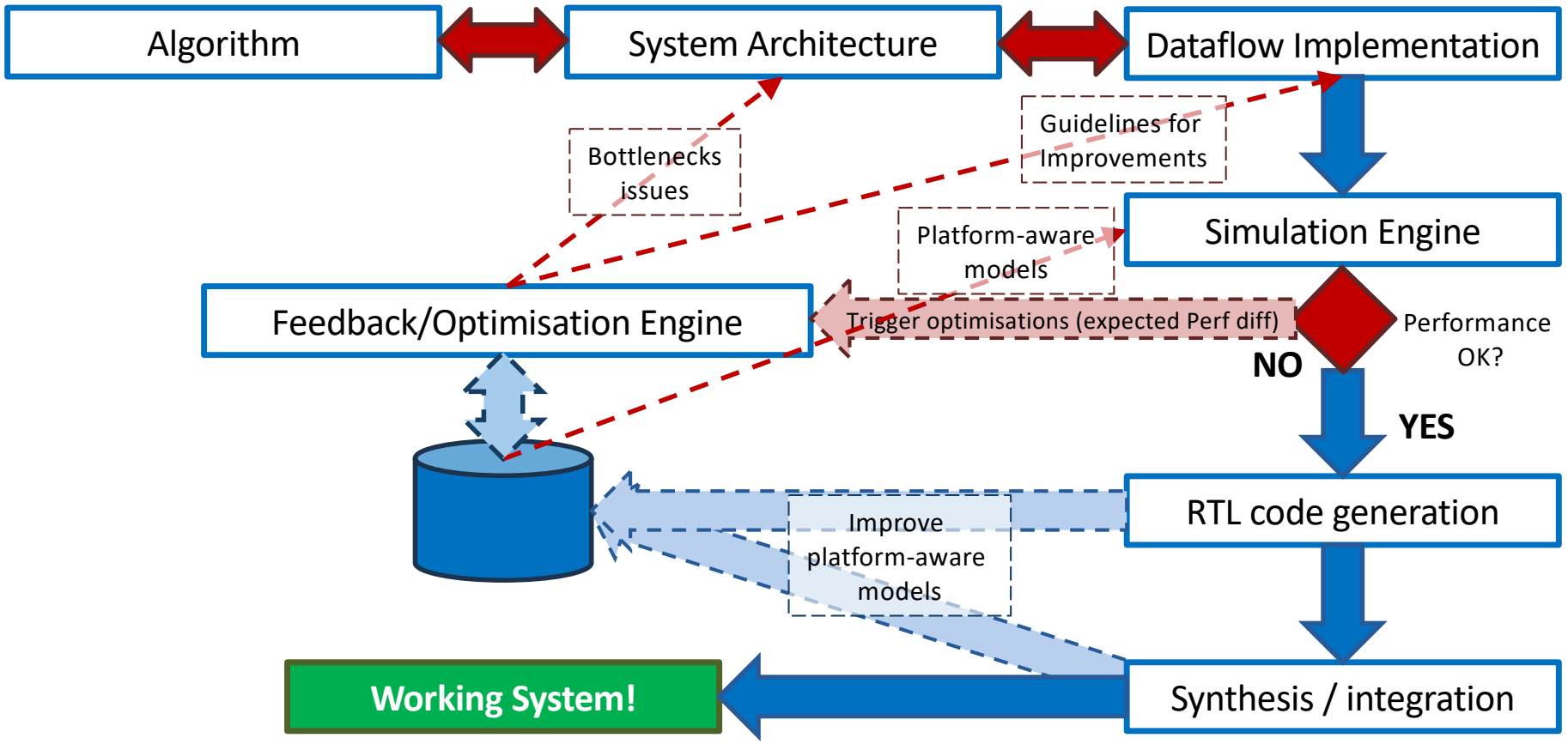
3. Stream data through the CG Custom Accelerator

Thinking Vertically about Solving Problems





Flow for Cross-layer Accelerator Development



Can this work well? (gzip on StratixV)[1]

In 2017 gzip in MaxJ was developed by **one intern** within one month (just software experience and one week of Max tutorials)

	IBM* (Verilog) (n=16)	Altera (OpenCL) (n=16)	MaxJ (n=16)	MaxJ (n=20)
Logic Utilization	45%	47%	42.8%	51.1%
BRAM	45%	70%	59.2%	88.6%

	Intel (i5 650 CPU)	IBM (Verilog)	Altera (OpenCL)	MaxJ (n=16)	MaxJ (n=20)**
Throughput	0.338 GB/s	**3.22 GB/s**	3.05 GB/s	3.2 GB/s	5 GB/s
Speedup	0.1x	**1x**	0.9x	1x	1.6x
Compression Ratio	2.18	**2.17**	2.17	2.25	2.27

*Based on Altera estimations of the post place and route image provided by IBM

** in addition to the four additional bytes/cycle also we pushed the frequency to 250MHz

[1] Nils Voss, Tobias Becker, Oskar Mencer, Georgi Gaydadjiev: *Rapid Development of Gzip with MaxJ*. ARC 2017: 60-71

12

Conclusions

Making FPGA based accelerators truly programmable? Why?

- YES, this is possible as shown before
- YES, it requires collaborative work and dedication
- YES, as LLMs/AI are not solution for all problems
- YES, as transistors became few atoms “wide”
- YES, because Quantum Computers keep “escaping us”

Questions

?